Intel 82374/5 EB/SB EISA Bridge PCIset Specification Update

October 1997

Order Number 297735-003

The Intel 82374EB/SB and 82375EB/SB EISA Bridge may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are documented in this Specification Update.

CONTENTS

REVISION HISTORY

Date of Revision	Version	Description
May 1996	-001	Initial Release
March 1997	-002	1. Adds ID Register contents change for LSI conversion 2. Adds 82375 EB/SB PCEB Specification Clarification: Ground Bounce on BE[3:0]. 3. Removes 82375 EB/SB PCEB Documentation Change for Special Cycle Enable (SCE) bit in the PCI Command Register.
October 1997	-003	Documentation Changes #1 and #2 have been moved into the Specification Changes section. Conversion to new template.

PREFACE

This document is an update to the specifications contained in:

82420/82430 PCIset EISA Bridge data book order number 290483-004 containing 82374EB/SB EISA System Component (ESC) data sheet (order number 290476) and 82375EB/SB PCI-EISA Bridge (PCEB) data sheet (order number 290477).

It is intended for hardware system manufacturers and software developers of applications, operating systems, or tools. It contains Specification Changes, Errata, Specification Clarifications, and Documentation Changes, and is divided into the following two parts:

- Part I: Specification Update for 82375 EB/SB PCEB
  Part II: Specification Update for 82375 EB/SB PCEB

Nomenclature

Specification Changes are modifications to the current published specifications. These changes will be incorporated in the next release of the specifications.

Errata are design defects or errors. Errata may cause the 82374EB/SB ESC or 82375EB/SB PCEB's, behavior to deviate from published specifications. Hardware and software designed to be used with any given stepping must assume that all errata documented for that stepping are present on all devices.

Specification Clarifications describe a specification in greater detail or further highlight a specification's impact to a complex design situation. These clarifications will be incorporated in the next release of the specifications.

Documentation Changes include typos, errors, or omissions from the current published specifications. These changes will be incorporated in the next release of the specifications.

General Considerations include system level considerations that the system designer should account for when developing hardware or software products using the 82374EB/SB ESC or 82375EB/SB PCEB.

S-Specs are temporary exceptions to the published standard specifications

Component Identification via Programming Interface(82374EB/SB ESC)

The 82374EB/SB ESC or 82375EB/SB PCEB's may be identified by the following register contents:

PCI Register	PCI Offset	Value
Vendor ID	N/A	N/A
Device ID	02h	00h
Revision Number	08h	02h (A-2 stepping) 03h (B-0 stepping) 13h (B-0 stepping, LSI Conversion)

Component Identification via Programming Interface(82375EB/SB PCEB)

PCI Register	PCI Offset	Value
Vendor ID	00h	8086
Device ID	02h	0482h
Revision Number	08h	03h (A-2 stepping) 04h (B-0 stepping) 05h (B-1 stepping) 15h (B-1 stepping), LSI Conversion

Part I:

Specification Update for 82374EB/SB ESC

GENERAL INFORMATION

This section covers the 82374EB/SB ESC.

Component Markings

82374EB/SB ESC COMPONENT MARKING INFORMATION

Stepping	S-Spec	Top Marking	Freq.	Notes
A-2	SZ867	S 82374EB, SZ867	33	No longer available
B-0		S 82374SB	33	Production

Summary Table of Changes

The following table indicates the Specification Changes, Errata, Specification Clarifications or Documentation Changes, which apply to the listed 82374EB/SB ESC steppings. Intel intends to fix some of the errata in a future stepping of the component, and to account for the other outstanding issues through documentation or Specification Changes as noted. This table uses the following notations:

CODES USED IN SUMMARY TABLE

X:	Erratum, Specification Change or Clarification that applies to this stepping.
Doc:	Document change or update that will be implemented.
Fix:	This erratum is intended to be fixed in a future stepping of the component.
Fixed:	This erratum has been previously fixed.
NoFix	There are no plans to fix this erratum.
(No mark) or (Blank Box):	This erratum is fixed in listed stepping or specification change does not apply to listed stepping.
Shaded:	This item is either new or modified from the previous version of the document.

82374EB/SB ESC

NO.	A2	B0	Plans	SPECIFICATION CHANGES	Documentation Update Status
1	X		Doc-NoFix	Digital Output Register must be programmed before DMA writes to Flash BIOS.	
2	X	X	Doc	Icc Specification defined.	
3		X	Doc	Some signals change from 3.3V friendly to 5V.	
4	X	X	Doc-NoFix	AC Timings Specifications.	
NO.	**A2**	**B0**	**Plans**	**ERRATA**	**Documentation Update Status**
1	X		Doc	SLOWH does not function properly.	
2	X	X	Doc-NoFix	Possible de-assertion of INTR for one clock.	
3	X	X	Doc-NoFix	Reset of Fail Safe Timer NMI.	
4	X	X	Doc-NoFix	EISA Configuration RAM Data.	
5	X	X	Doc-NoFix	Using the 82C54 Timers.	
NO.	**A2**	**B0**	**Plans**	**SPECIFICATION CLARIFICATIONS**	**Documentation Update Status**
1	X	X	Doc	Reading and Writing Configuration Registers.	
2	X	X	Doc	Configuring Divide By Three Mode.	
3		X	Doc	Interrupt Steering Programming Considerations.	
NO.	**A2**	**B0**	**Plans**	**DOCUMENTATION CHANGES**	**Documentation Update Status**
1				There are currently no known documentation changes.	
NO.	**A2**	**B0**	**Plans**	**GENERAL CONSIDERATIONS**	**Documentation Update Status**
1		X	Doc	INTR & “Thru Local Mode”.	
2		X	Doc	Pulsing of APICD1 (APCIEN) during CPU Reset (CPURST).	
3		X	Doc	Interrupt Levels & System Event Generation in Power Managed Systems.	
4		X	Doc	APICCLK Signal Quality Issues.	
5		X	Doc	APIC Bus Layout	

82374EB/SB ESC SPECIFICATION CHANGES

1. DMA Writes to Flash BIOS

Before attempting to perform DMA writes to Flash BIOS from a floppy controller not located on the X-Bus, a '0' must be written to the Digital Output Register first. Otherwise, contention will occur on the data bus.

2. ICC Specification Defined

4 EISA Slots : Icc (max) = 120mA

8 EISA Slots : Icc (max) = 150mA

3. Change from 3.3V Friendly Signals to 5V Signals on B-0 Step

The following signals INT, NMI, IGNNE#, SMI#, ALTA20, STPCLK#, and ALTRST# were changed starting with the B-0 step to 5V signals.

These signals, with the exception of APICD[1:0], are 5 volt signals in the B-0 Step of the ESC with a IOL of 2 ma @ 0.2 volts. The APICD[1:0] lines are open-drain and are typically pulled up to 3.3 volts with a resistor. These signals will have to be translated to 3.3 volts, depending on whether they are driven directly to the processors or are already gated with other signals, e.g. ALTA20. As a 5V-to-3.3V example: an open-collector buffer can convert the 5 volt A20M- signal to the A20MLV- via pullup resistor to 3.3 Volts. The output signals noted here are driven to a logical low during system reset (while PCIRST# is asserted).

4. AC Timing Specifications

PROBLEM: The following AC Timing Specifications are changed for the B-0 ESC:

Symbol	Parameter	Original Value	New Value
t8b	MASTER16# hold from BCLK rising	0.00	1.60
t16b	M16# hold from BCLK falling	0.00	1.20
t19b	EXRDY# hold from BCLK falling	0.00	2.50

IMPLICATION: No known negative system implications

WORKAROUND: None required

STATUS: Not fixed on B-0 Step, A-2 workaround continues to apply to B-0 Step.

82374EB/SB ESC ERRATA

1. SLOWH# Function Does Not Operate Correctly

PROBLEM: Upon the falling edge of RESET, SLOWH# is asserted.

IMPLICATION: If SLOWH# is tied to the processor's HOLD input, the processor will not be able to begin fetching code, and system failure will result.

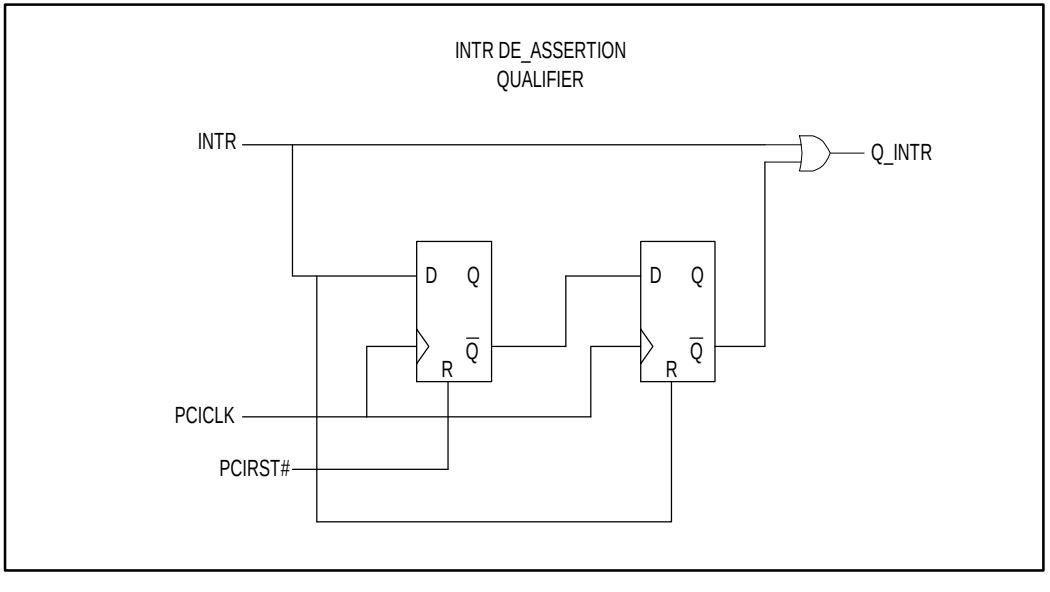
WORKAROUND: Do not use the SLOWH# function.

2. Interrupt De-assertion

PROBLEM: While the ESC is driving INTR to the processor, and PEREQ# is asserted followed by either NMFLUSH# or EISAHLDA going low (for example when ownership of the EISA bus is given to the PCEB, or during interrupt acknowledge cycles), INTR will be de-asserted for one PCI clock.

IMPLICATION: Possible de-assertion of INTR for one PCI clock. This may generate spurious interrupts and can be potentially reported as “lost interrupts” under certain operating systems such as Novell.

WORKAROUND: A workaround has been provided which will allow de-assertion of INTR only if INTR is low for at least two PCI clocks.



3. Reset of Fail-Safe Timer NMI

PROBLEM: The Fail-Safe Timer NMI is not completely cleared by writing a 0 to bit 2 of port 0461h, it is only masked.

IMPLICATION: As soon as the Fail-Safe Timer NMI is enabled, an NMI will be generated, even if the timer has not yet been started. After the timer has expired, the NMI is masked by writing a 0 to bit 2 of port 0461h (the Fail-Safe Timer control bit), but an NMI will re-assert if bit 2 of port 0461h is set back to 1.

WORKAROUND: The output from the Fail-Safe Timer (Timer 2, Counter 0, Mode 0) must be cleared by writing a control word or timer value to it before the NMI is enabled. For example, this can be done by writing a 20h to 4Bh before setting bit 2 of port 0461h to a 1 to enable the Fail-Safe Timer NMI.

4. EISA Configuration RAM Data

PROBLEM: EISA configuration RAM data. The 82374EB may not generate the proper page address on the LA[31:27]# lines during configuration RAM accesses under the following conditions: A DMA cycle or ISA refresh cycle, followed by another pending DMA, followed by an access to EISA configuration RAM (an EISA master should create the same problem).

IMPLICATION: During the configuration RAM reads, the 82374EB decodes the cycle properly and generates the signals necessary for the access. The system tries to access page xxxxx and program the proper page address into the C00h register. The 82374EB should drive xxxxx on LA[31:27]# at this time, but instead these address lines may float under the failing scenario. These address lines float when NMFLUSH# (New Master Flush) is asserted. NMFLUSH# gets asserted because of the back to back masters (EISA/ISA/refresh) being serviced.

Under normal conditions such as reading and writing to EISA configuration RAM before running application software there is no issue. The problem may occur when accessing EISA configuration RAM with multiple bus mastering events pending.

WORKAROUND: Below are two possible solutions:

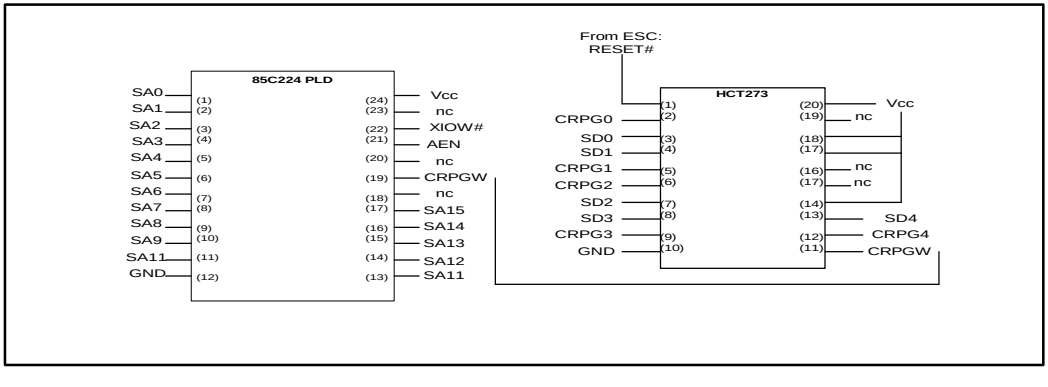


Figure 2: PLD Workaround #1

1. The 85C224 PLD in fig. 2 is programmed to generate an IO write to I/O port 0C00h, which the ESC integrates as a configuration RAM page register. Upon seeing 0C00h on the address bus (SA[15:0]), AEN

asserted high for EISA bus master cycles, and an XIOW# asserted low, the PLD will drive its only output, CRPGW asserted high, as the clock signal input for the HCT273.

The reason for the driving the output high is that the HCT273 needs to see a rising edge on this clock signal in order to latch the correct CRAM page address. The HCT273 is an Octal D-type flip flop which is enabled on the rising edge of its clock signal input. The HCT273 will latch the correct CRAM page address off the data bus (SD[4:0]) and drive the address on its output signals (CRPG[4:0]). CRPG[4:0] are the synthesized versions of LA[31:27] and are tied to the corresponding inputs on the NVRAM. This solution is transparent to the system BIOS.

The actual pin assignments are given in fig. 2 above. This also includes the asserted values of the inputs and outputs (/ = symbol for asserted low).The following is the PLD equation (written in PLDshell Plus) for the data setup of the correct CRAM page address:

```
EQUATIONS
/CRPGW   =    /SA15 * /SA14 * /SA13 * /SA12 *
                    SA11 * SA10 * /SA9 * /SA8 *
                    /SA7 * /SA6 * /SA5 * /SA4 *
                    /SA3 * /SA2 * /SA1 * /SA0 *
                    XIOW * AEN
```

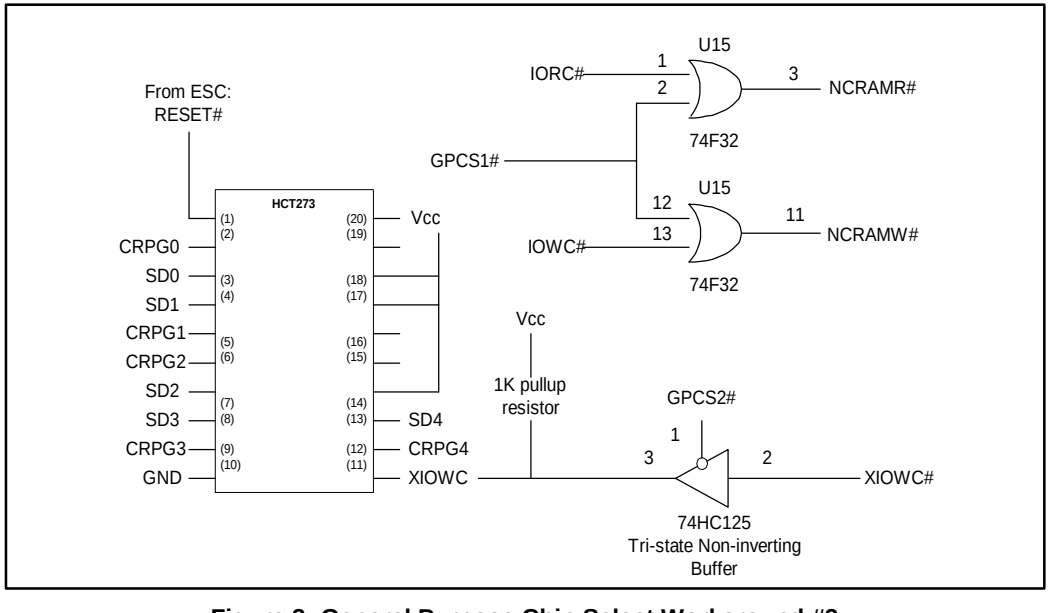


Figure 3: General Purpose Chip Select Workaround #2

1. This workaround uses the General Purpose Chip Select outputs available from the ESC. The first signal is GPCS1# (pin 159 on the ESC) and the second signal is GPCS2# (pin 160). Software changes will have to be made to the ESC Configuration Registers in order to use the General Purpose Chip Selects. The system BIOS will need to be modified in order to use this solution.

   GPCS1# will be OR'd with IORC# and IOWC# to synthesize the NVRAM configuration reads and writes, NCRAMR# and NCRAMW#. These two signals will be connected to the NVRAM (pins 22 and 27), respectively. The pins on the NVRAM will be lifted in order to connect the new synthesized signals. In addition, the CRAM decode will be disabled as part of the changes to the ESC configuration registers. The OR-gates will be supplied by a 74F32 that is already populated on the 82430NX DP-EISA board, device U15. The IORC# and IOWC# signals will be taken of the device U12, pins 2 and 4, respectively.

   GPCS2# will be used to enable a 74HC125 buffer. This buffer will use XIOWC# from device U12 (pin 16 on the 82430NX board), and route it to the HCT273 (pin 11). This will be the clock signal that will enable the HCT273 Octal D-type flip flop on a rising edge, thus giving the correct CRAM page address. The HCT273 will latch the correct CRAM page address off the data bus (SD[4:0]) and drive the address on its output signals (CRPG[4:0]). CRPG[4:0] are the synthesized versions of LA[31:27] and are tied to the corresponding inputs on the NVRAM. The 1k pull-up resistor is used to keep the signal high when the buffer is not enabled.

   Note: The rising edge of the XIOWC# signal meets the required hold time of the data. This timing also takes into account the flight time, the propagation delay through the buffer, and the propagation delay of the buffered version of the IOWC# signal. It is recommended that you use a 74ACT125 buffer which will give a wider margin in meeting the timing specifications.

   The following assembly language code should be implemented, as part of the initialization of the chip set at boot time, within the BIOS. This code makes the necessary changes in the ESC Configuration registers in order to enable the desired General Purpose Chip Selects.

General Purpose Chip Select Code

```
mov dx,22h          ;Allow access to ESC configuration registers
mov al,02
out dx,al

mov dx,23h
mov al,0fh
out dx,al

mov dx,22h          ;Disable Configuration RAM Page address
mov al,40h          ;(CPG[4:0]) generation--bit5=0
out dx,al           ;Enable GPCS[2:0] functionality--bit4=0

mov dx,23h
mov al,44h
out dx,al

mov dx,22h          ;Disable CRAM decode which also disables the
mov al,4fh;the generation of CRAMRD# and CRAMWR# signals bit7=0
out dx,al
```

```
mov dx,23h
mov al,7fh
out dx,al

mov dx,22h          ;Set the General Purpose Peripheral mapping address
mov al,6ch          ;GPCS2# Low Address
out dx,al

mov dx,23h
mov al,00h
out dx,al

mov dx,22h          ;GPCS2# High Address
mov al,6dh
out dx,al

mov dx,23h
mov al,0ch
out dx,al

mov dx,22h          ;GPCS2# Mask Register
mov al,6eh
out dx,al

mov dx,23h
mov al,00h
out dx,al

 mov dx,22h         ;GPCS1# Low Address
mov al,68h
out dx,al

mov dx,23h
mov al,0ffh
out dx,al

mov dx,22h          ;GPCS1# High Address
mov al,69h
out dx,al
```

```
mov dx,23h
mov al,08h
out dx,al

mov dx,22h          ;GPCS1# Mask Register
mov al,6ah
out dx,al

mov dx,23h
mov al,0ffh
out dx,al

mov dx,22h          ;Enable XBUSOE# generation for GPCS1#--bit1=1
mov al,6fh
out dx,al

mov dx,23h
mov al,0fah
out dx,al
```

5. Using the 82C54 Timers

Two issues have been uncovered when using the 82C54 timers which are integrated into the 82374SB. In the PC-AT architecture, the three timers in the 82C54 perform the functions listed below.

- Timer 1. System timer interrupt
- Timer 2. Refresh request
- Timer 3. Speaker tone

The two known issues are attributed to the fact that the clock used to read and write the 82C54 (PCI Bus Clock) and the 82C54 timebase clock (14.316 MHz OSC) are asynchronous to each other.

PROBLEM 1: Reading incorrect timer/counter data.

IMPLICATION: Because of the asynchronous clocks, the timers in the 82C54 cannot always be read reliably. Intermittently, the timer is read at the same time the timers count is changing and an incorrect value is latched. The 82C54 read back and counter latch commands are also affected.

WORKAROUND: Perform repeated reads checking for same value.

PROBLEM 2: Timer does not begin counting on rising edge of GATE in timer mode 1, for timer 2.

IMPLICATION: This mode uses a rising edge on the timer GATE input to start the timer. The timer GATE input can be toggled by writing to I/O port 61H. Because the GATE pin is asserted relative to PCICLK, the setup and hold time requirements of the GATE pin relative to the 82C54 OSC clock can be violated. Because GATE is edge triggered in timer mode 1, the 82C54 does not recognize the rising edge and the timer does not begin counting. In the PC-AT architecture, timer 2 is the only timer that has its GATE input connected to an I/O port. All the other GATE inputs are tied high. This issue was uncovered during 82C54 diagnostic testing.

WORKAROUND: None needed.

82374EB/SB ESC SPECIFICATION CLARIFICATIONS

1. Reading and Writing Configuration Registers

When changing values in configuration registers which contain one or more reserved bits, care must be taken to avoid changing the value of the reserved bits. It is not safe to assume that because a bit is reserved that it has no function or meaning. The proper procedure for changing a register value is to first read the register, change only the required bits (without changing the value of any reserved bits), then write the new value back out to the register.

2. Configuring Divide By Three Mode

Care must be taken when changing the EISA Clock Divisor register (ESC address offset 4Dh) from the default clock divisor mode of divide by 4 to divide by 3 because of the protocol used to keep the PCEB and the ESC synchronized. The following steps are necessary:

a) Open the configuration space of the ESC (index address-data registers at I/O locations 22 and 23h).

b) I/O write to 4Dh (clock divisor register) to I/O location 22h (index address register).

c) Immediately write the divide by 3 enable bits [2:0] to I/O location 23h (index data register). Do NOT read the contents of 23h before writing the bits (a read modify write operation may cause the BIOS to read only the value in the ESC and not the PCEB).

If it is necessary for BIOS to read the contents of the clock divisor register before changing the clock divisor do the following:

- I/O write to 22h with 4Dh.

- I/O read from 23h (to examine the value in the register).

- I/O write to 22h with 4Dh again.

- I/O write to 23h with the desired clock divisor bits [2:0].

** These I/O writes and reads MUST be byte transfers only and cannot be word transfers.

3. Interrupt Steering Programming Considerations

When using the PCI programmable interrupt steering feature in the ESC, the following programming considerations apply:

1. Any interrupt which a PIRQx# is steered to must be programmed to level sensitive in order for the interrupt controller to recognize it.

2. For an interrupt used as a PIRQx#, that IRQ pin is also level sensitive. It is not permissible to use an interrupt on the ISA bus as edge triggered as well as on the PCI bus as level sensitive.

3. Registers which must be programmed when using a PIRQx# include (in suggested order):
   - Mode Select Registers (ESC offset 40h bits 6 and 1:0)
   - Edge Level Registers (I/O 4D0h and 4D1h)
   - PIRQ[3:0]# Route Control Registers (ESC offset 63:60)
   - Interrupt Mask Registers (I/O 21h and A1h)

82374EB/SB ESC DOCUMENTATION CHANGES

There are currently no known documentation changes.

82374 EB/SB ESC GENERAL CONSIDERATIONS

1. INTR and "Through Local Mode"

The B-1 through B-5 steppings of the Pentium® processor at ICOMPTM index 735\90 (or 815\100) have an errata (10AP) when using the local APIC in Through Local Mode (virtual wire). There are some software workarounds proposed in the CPU stepping information. The following is a hardware workaround which allows the customer to still use the local APIC in Through Local Mode.

If a processor has its Local APIC setup in Through Local Mode (virtual wire), the 82374SB can potentially deassert and then reassert INTR before the interrupt acknowledge cycle for the first interrupt completes. When this happens, the CPU ignores the second interrupt and the system eventually hangs because no further interrupts will be serviced.

The CPU's local APIC does not recognize that an interrupt acknowledge has occurred until BRDY# is returned to the CPU for the second interrupt acknowledge bus cycle on the host bus. The 82374SB will deassert INTR soon after the FRAME# occurs on the PCI bus for the interrupt acknowledge cycle. If another interrupt occurs and INTR reasserts before BRDY# gets returned to the CPU by the 82434NX PCMC (or any Host to PCI bridge component), the interrupt will not be recognized.

Analysis of the 82434NX PCMC during an interrupt acknowledge cycle shows that a maximum of 106 HCLKs can occur between FRAME# and BRDY# being asserted on the host bus. The ESC can reassert INTR as soon as 74 host clocks after receiving FRAME# of the interrupt acknowledge cycle. There is a "window" of 32 HCLKs in which INTR can be reasserted by the ESC.

One workaround could use a combinatorial circuit using LOCK# and MI/O# from the processor to block the assertion of INTR until the interrupt acknowledge cycle has completed. Loading and timing on the LOCK# and MI/O must be considered for this workaround as they are host bus signals. The following would be the logic equation used for INTR:

INTR_OUT = ((INTR * INTR_OUT + (INTR*M/IO#) + (INTR*LOCK#))

Another workaround can be implemented by delaying any assertion of INTR by 32 host clocks. An RC circuit on the output of the required external 3.3V translation buffer (required for the B0 stepping of the ESC) signal can also accomplish this delay. The 32 host clocks is only valid for systems which use the 82434NX host bridge.

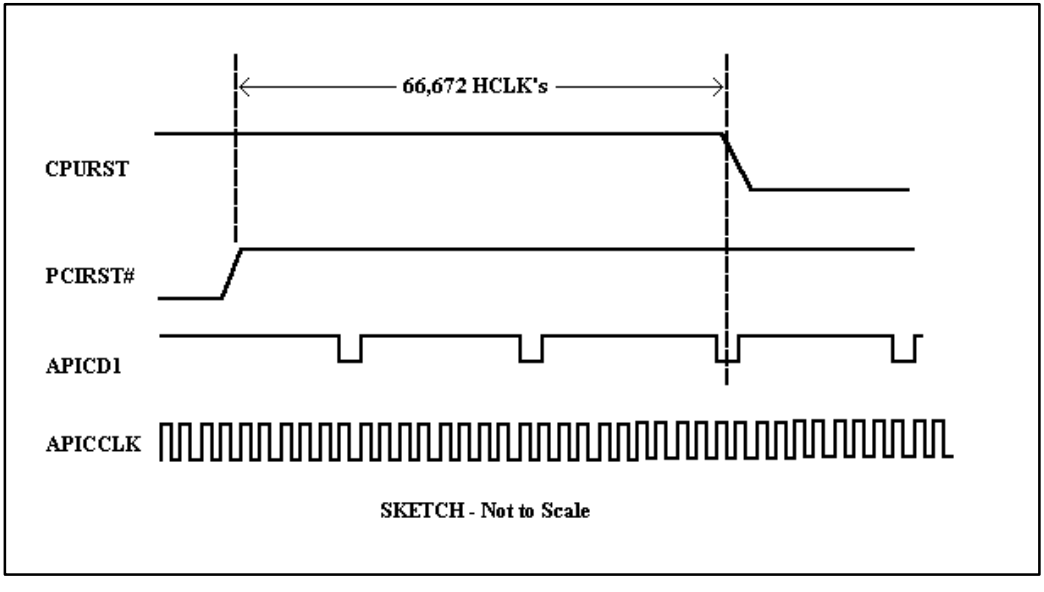
2. Pulsing of APICD1 (APICEN) during CPU RESET (CPURST)

This issue affects dual-processing (DP) EISA systems using the Pentium® processor at ICOMP index 735\90 (or 815\100) and the A-1 stepping of the 82434NX PCMC host bridge. Single processor designs are not affected.

The CPUs APIC data lines, APICD0 and APICD1, have secondary functions when CPU RESET (CPURST) is active. APICD1 is the APICEN (APIC enable) pin while APICD0 is the DPEN# (dual processor enable) pin. The local APICs are enabled if APICEN is sampled active high on the falling edge of CPURST. Also during RESET, the second CPU will drive DPEN# active low to indicate to the boot processor that it is present.

On the A-1 82434NX PCMC, there is approximately a one millisecond delay (specifically 66,672 external host clocks) between the deassertion of PCIRST# and the deassertion of CPURST following a cold system boot (i.e. the PWROK input of the PCMC transitions from low to high). During this one millisecond window, the I/O APIC in the ESC, which is no longer in reset, begins to monitor the APIC bus for valid messages. APICEN (APICD1 = logic 1) and DPEN# (APICD0 = logic 0) remain asserted during this period. The I/O APIC interprets the states of these two bits as the beginning of a normal APIC message and continues to sample the APIC data lines. The I/O APIC responds, per the APIC protocol, by driving APICD1(APICEN) low once every 20 APICCLK's indicating a checksum error has been detected on the current message. Under these circumstances, this is the correct behavior for the ESC.

From the time that PCIRST# deasserts until the time that CPURST deasserts, the I/O APIC periodically pulses APICD1 low for one out of every 20 APICCLK's. If the pulse on APICD1 (APICEN) aligns with the falling edge of CPU RESET (CPURST), the local APIC's will not be enabled; this prevents proper DP or DP-ready functionality. See the figure below.



Hardware Workarounds:

Workaround #1: Calculate the window for the APICD1 pulse relative to CPURST for a given host clock (HCLK) and APIC Clock (APICCLK) frequency. The following example demonstrates that by using a 60.0 MHz HCLK and a 16.0 MHz APICCLK (0.01% accuracy for both) the APICD1 pulse will be outside the sampling window of CPURST. With these equations, equally acceptable windows can be found for other values of HCLK and APICCLK.

Note that with the above frequency accuracy, a 16.0MHz APICCLK with a 50MHz or 66MHz HCLK may not always function correctly under worst case clock variations in a DP EISA environment.

Sample Calculation:

Three key variables in the system first must be identified:

I. The accuracy of the host clock oscillator input to the PCMC (HCLKOSC).
- Additional inaccuracy in HCLK or PCI Clock (PCLK) is not induced by the internal PCMC PLL's.

II. The accuracy of the APIC clock oscillator.

III. An internal version of PCIRST# (referred to as iPCIRST#) defines when the I/O APIC exits reset.

- iPCIRST# is deasserted on the fourth rising edge of BCLK after PCIRST# is sampled inactive.

- Depending upon the PCLK/BCLK alignment, this window ranges from 14-18 PCLK's.

The following example assumes a 60MHz Host Clock (HCLK) frequency with an oscillator accuracy of 0.01% and a 16.0MHz APIC Clock (APICCLK) frequency with an accuracy of 0.01%.

STEP 1: Determine the nominal clock frequency for HCLK, PCLK and APICCLK. Also, determine the accuracy for HCLK and APICCLK. Since PCLK is HCLK/2, no additional inaccuracy is incurred.

	Frequency	Accuracy
HCLK	60.000 MHz	0.01%
APICCLK	16.000 MHz	0.01%

STEP 2: Subtract the minimum number of HCLKs (28) for iPCIRST# to deassert from 66,672 HCLKs. This is the maximum time window that the APICD1 pin will be pulsing until the falling edge of CPURST. The I/O APIC is no longer in reset and begins to monitor the APIC bus for valid messages during this maximum window. The APIC data pins meanwhile remain in their configuration mode until the falling edge of CPURST.

Conversely, subtract the maximum number of HCLKs (36) for iPCIRST# to deassert from 66,671 HCLKs. One less HCLK was used to account for the uncertainty between HCLK and PCLK. This provides the minimum time window between the I/O APIC responding to the APIC messages until the falling edge of CPURST.

Convert these two extremes to absolute time (seconds). When converting the max HCLK window to real time be sure to use the max HCLK Period possible due to clock tolerances. Similarly, use the min HCLK period for the min HCLK window.

Equation #1:

(HCLK cycles -Min iPCIRST cycles) * Max HCLK Period = Max HCLK Window

(66,672 - 28) * (1.66683335E-08) = 1.11084442 mS

(HCLK cycles - Max iPCIRST cycles) * Min HCLK Period = Min HCLK Window

(66,671 - 36) * (1.66650002E-08) = 1.11047229 mS

	HCLKs	APICD1 Pulse Window (s)
Max HCLKs - Min iPCIRST# time	66,644	1.11084442E-03
Min HCLKs - Max iPCIRST# time	66635.00	1.11047229E-03

STEP 3: Determine the equivalent number of APICCLKs contained in the maximum and minimum pulse windows calculated above. Account for the accuracy of the APICCLK source.

Also, since APICCLK and HCLK are completely asynchronous to each other, there potentially could be a one APICCLK reduction in the pulse window if the iPCIRST# deassertion happens to align with the rising edge of APICCLK. Therefore, one APICCLK should be subtracted from the minimum window to account for any HCLK/APICCLK misalignment.

Equation #2:

(Max APICD1 pulse window) * (Max APICCLK frequency) = Max APICCLKs

1.1108444 mS * 16.0016 MHz = 17,775.29 APICCLK's

(Min APICD1 pulse window) * (Min APICCLK frequency) - 1 = Min APICCLKs

1.11047229E-03 * 15.9984 MHz - 1 = 17,764.78 APICCLKs

APICCLK Frequency	Max window (APICCLKs)	Min Window (APICCLKs)
16.0 MHz + 0.01%	17775.29	17768.33
16.0 MHz - 0.01%	17771.73	17764.78

STEP 4: Select the maximum and minimum number of APICCLKs contained within the pulse window. The two cases are underlined in the Step 3 calculation above. For each case, divide the number of APICCLKs by 20 to determine the minimum and maximum number of APICD1 pulses driven by the I/O APIC during this window. APICD1 is pulled to a logic 1 via the strong, external pullup resistor for 19 APICCLKs before pulsing low via the I/O APIC for 1 APICCLK.

	Max APICD1 Pulses	Min APICD1 Pulses
APICD1 (APICEN) Pulses	888.76	888.24

STEP 5: The fractional portion of each pulse calculation above provides the time (in APICCLK's) between the final APICD1 pulse and the falling edge of CPURST.

To calculate the APICEN setup time for the maximum pulse condition (888.76444 pulses), multiply the fractional portion of the pulse calculation (0.76444) by 20 and then divide by the maximum APICCLK frequency. This results in the actual setup time between the end of the last APICD1 pulse and the falling edge of CPURST. Subtracting this setup time from [(19 divided by the maximum APICCLK frequency)] results in the APICEN hold time from the falling edge of CPURST.

To calculate the setup and hold times for APICEN for the minimum pulse condition (888.23894), the minimum APICCLK frequency is used.

Equation #3 - Maximum APICD1 Pulse Condition

[(fractional number of max APICD1 pulses) * 20]/(max APICCLK frequency) = APICEN setup time

[(0.76444) * 20]/(16.0016 MHz) = 955.5 nS

[19/(max APICCLK frequency)] - APICEN setup time = APICEN hold time

[19/(16.0016 MHz)] - 955.5 nS = 231.9 nS

Equation #3 - Minimum APICD1 Pulse Condition

[(fractional number of min APICD1 pulses) * 20]/(min APICCLK frequency) = APICEN setup time

[(0.23894) * 20]/(15.9984 MHz) = 298.7 nS

[19/(min APICCLK frequency)] - APICEN setup time = APICEN hold time

[19/(15.9984 MHz)] - 298.7 nS = 888.9 nS

	APICEN Setup Time to CPURST falling (ns)	APICEN Hold Time from CPURST falling (ns)
Max APICD1 Pulses (888.76)	955.50	231.90
Min APICD1 Pulses (888.24)	298.70	888.90

STEP 6: If the whole number portion of the pulses calculated in Step 4 is identical in the minimum and maximum cases, then the minimum setup and hold time margins for APICEN easily fall out. In this example, the minimum setup time to the falling edge of CPURST is 298.7ns while the minimum hold time from the falling edge of CPURST is 231.9ns. Therefore, for this combination of oscillator frequencies and accuracy, worst case APICEN setup and hold times clearly meet the minimum CPU setup and hold specifications of 2 HCLKs. APICEN will be properly sampled by the CPUs under all circumstances.

If the whole number portions are not equal for the minimum and maximum cases, then the accuracy of the HCLK and APICCLK frequencies is not sufficient to keep the APICD1 pulse from aliasing, i.e. shifting through the entire 20 APICCLK window under worst case conditions. Choose frequency sources with better accuracy.

Workaround #2: Add logic to ensure that PCIRST# remains asserted low until PWROK is valid and then once PWROK is valid, let PCIRST# become the logical inverse of CPURST. This ensures that the I/O APIC and both CPU's exit reset simultaneously thereby preventing the I/O APIC from responding to the phantom APIC messages.

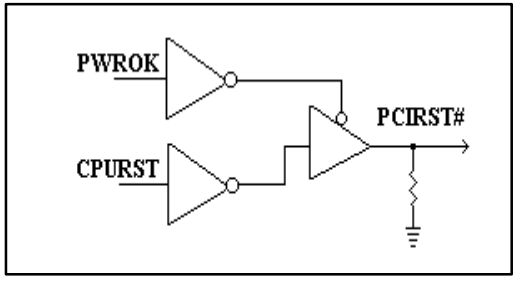
Possible Implementations:

a) Connect a buffered version of CPURST to an inverter and then connect the output of this inverter to the input of a three-state buffer. PWROK# is used as the active low enable to the three-state buffer. The output of this three-state buffer is used as the PCIRST# signal for the entire system. Note that the PCMC PCIRST# output pin (pin 147) should be disconnected from the system.

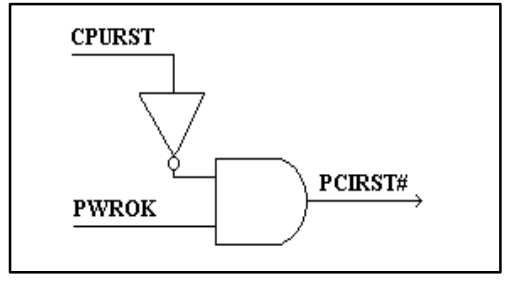
It is important to use a buffered version of CPURST and preferably one that already exists in the system due to the critical AC timing of this signal. A weak pulldown resistor (8.2K, for instance) should be added to the PCIRST# net to ensure it is low while PWROK is low.

b) Connect a buffered version of CPURST to an inverter and then connect the output of this inverter to one input of a two-input AND gate. PWROK is the second input to the AND gate. The output of the AND gate becomes PCIRST# for the entire system. Again, the PCMC PCIRST# output pin (pin 147) should be disconnected from the system.

Workaround "A"



Workaround "B"



NOTE: Due to the combinational logic in workaround "b", there is a short time period (potentially several HCLKs) between the assertion of PWROK and the assertion of CPURST when PCIRST# actually deasserts. This deassertion of PCIRST# does not cause any system problems since it takes at least several hundred nanoseconds before any signals in the ESC will respond.

Design Considerations:

On the B-0 stepping of the ESC (82374SB) but not the A-2 stepping (82374EB), workaround #2 will force the INIT signal to be sampled high by the boot processor on the falling edge of CPURST. This causes the boot processor to enter BIST (Built in Self Test). Consequently, the first ADS# from the boot processor will be delayed by approximately 219 core clocks (the duration of the Built In Self Test) before continuing in normal operation.

The system designer has the option of gating the INIT input low until CPURST is sampled inactive if entering BIST mode is not desired. The system designer should ensure that the appropriate CPU INIT setup and hold times are met if this approach is taken.

Workaround #3: Gate the APICCLK with the falling edge of CPURST. The APICCLK can be disabled while CPURST is active either selectively to only the I/O APIC or to all APIC agents. Without an APIC Clock, the I/O APIC is prevented from sampling the APIC data bus until after the falling edge of CPURST. Consequently, the pulsing of the APICEN signal will not occur.

3. Interrupt Levels and System Event Generation in Power Managed Systems

The 82374SB can use the 14 IRQ input pins (IRQ1,IRQ3:7,IRQ8#,IRQ9:15) to generate hardware system events in PCI systems that support power management mode by programming the System Event Enable Register, bits 15:0, corresponding to the selected interrupt. Detection of these events will cause the Fast Off Timer to be re-loaded with its initial count value and deassert the STPCLK# pin. The logic inside the SIO samples the enabled interrupts as level sensitive, active high signals, for system event determination.

Most motherboard devices or ISA add-in cards which use an interrupt will drive the interrupt low when the interrupt is inactive, and only drive the interrupt high when it needs to generate an interrupt to the CPU. These devices will work properly with the ESC level sensitive logic. A device or ISA add-in card that does not drive its interrupt low when inactive should not be used with the ESC although the following description provides a way to handle this type of interrupt functionality.

This type of device or ISA add-in card floats the interrupt signal when not actually generating an interrupt which allows the motherboard to pull the interrupt signal high. The ESC will interpret the high value of the interrupt signal as a break event (when enabled) causing the Fast Off Timer to

be continuously re-loaded. As a result, the Fast Off state is never entered into, SMI is not generated, STPCLK# is not asserted, and power savings can't be realized. The following description will refer to this type of device or ISA card as a 'floating interrupt' device or ISA card.

A floating interrupt device or ISA card can be made to work properly in a system with power management enabled by not recognizing its interrupt as a system event, provided it can tolerate a long interrupt latency period. Since the CPU is clock throttled using the STPCLK# signal, the interrupt from this type of device will still get serviced during the STPCLK# inactive period and function properly.

As an example, if a floating interrupt card is installed in the system, and the clock throttle STPCLK# low and high timers are programmed as below,

Register name	Config Offset	Value	Comment
Clock Throttle STPCLK# Low Timer	ACh	3	STPCLK# low for 96 uS
Clock Throttle STPCLK# High Timer	AEh	1	STPCLK# high for 32 uS

then the floating interrupt card or device would need to be able to buffer 96usec worth on incoming and transfer that data to the system in less than 32 uS.

A 14400 baud modem could transfer up to 1500 bytes/sec and only transfers 0.14 byte in 96uS so no buffering should be required.

Network and hard disk controller cards are divided into 3 major classes: non-intelligent buffered cards, DMA masters and ISA Masters. The DMA and ISA master cards can transfer data into memory whether the CPU is on STPCLK or not. The non-intelligent cards need enough buffering for up to 96uS of incoming data. Most non-intelligent network and hard disk controller cards contain a minimum of 4K bytes of RAM to buffer incoming data which is sufficient to buffer 96uS of incoming data. For example, a 10-Mbyte/sec SCSI bus can transfer a maximum of 960 bytes in 96uS.

4. APICCLK Signal Quality Issues

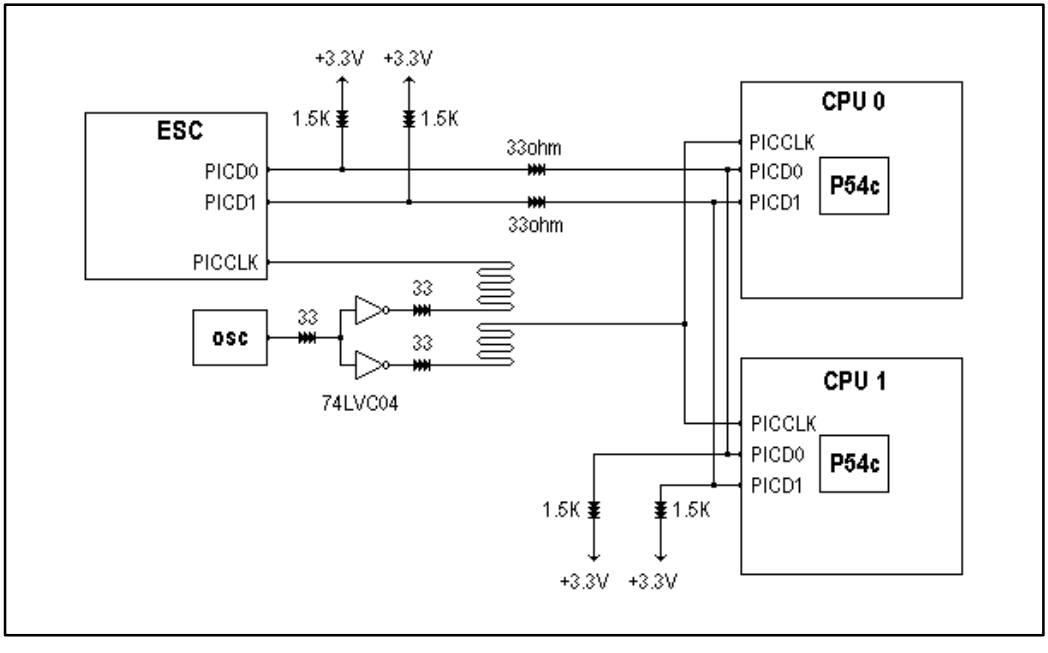
It has been found on several ISA and EISA dual processor boards that the Pentium® processors at iCOMP® Index (735\90MHz, 815\100MHz) are susceptible to signal quality issues on the APICCLK input**.** Designs should strive to make this clock as clean as possible at the input of both CPUs.

Further details can be found in the " Pentium processor at iCOMP Index 735\90 MHz Pentium processor at iCOMP INDEX 815\100 MHz B3-Step STEPPING INFORMATION" document, section 8AP. It covers this subject in some detail including routing and topology guidelines.

5. APIC Bus Layout

In addition to the signal quality susceptibility on the APICCLK input on the Pentium processor, there exists a similar issue with signal quality susceptibility on the Pentium processor's PICD0 and PICD1 data lines. In prior board layout designs, the APIC data lines were tied high to $V_{CC}3$ with single 150 - 250 ohm pull-up resistors located near the Pentium processors. This has been re-designed.

The new design layout implements a termination scheme on the APIC data lines where two pull-up resistors, each located on opposite ends of the trace and in parallel, are implemented to maximize signal noise margin with minimal rise time (setup) penalty on these open drain signals.



The following are general guidelines & recommendations for the APIC Bus Layout Scheme:

GENERAL: Minimize the trace length between CPUs (mount as close together as possible).

For maximum noise immunity (from cross coupling), separate APIC Bus signals from other signals on the board (esp. data bus) with at least 20 mil spacing.

Keep the resistor terminators as close to the end of the lines as possible (as close to the components as possible).

PICCLK: Minimize clock skew between CPUs and ESC (and/or others on APIC Bus) by equalizing APIC clock trace lengths.

Use a "T" routing scheme between CPUs with each leg being of equal length.

Use a series termination that closely matches the characteristic impedance of the trace, as described in more detail in the Pentium® processor Specification Update document (section II: Spec. clarification #3). ex., a 13 ohm resistor was implemented by a designer. Simulation and actual experimentation will help converge on the correct value for each specific design.

PICD0/D1: Use "straight line" routing from ESC to CPU1 to CPU2

Allow provisions for terminating trace at both ends. A pull-up resistor to Vcc3 should be placed as close to the ESC and CPU2 as possible. Through simulation and experimentation, the exact pull-up value can be selected (1.2Kohm - 1.5Kohm).

Provisions for a series termination (~33 ohms) at the midpoint of the APIC data line traces should be considered to alleviate any undershoot or reflection problems.

Since the APIC data lines are open drain signals, they require external pull-up resistors to achieve the logic level "1" state (i.e. Vih, min). Fundamentally, the stronger the pull-up, the faster the signal will reach a "1". However, the smaller pull-up resistor value will also raise the effective Vol for these signals (thereby reducing the noise margin to Vil).

Choosing the appropriate pull-up value for these signals is system dependent (with regards to noise immunity required, trace impedance, etc.). The finite boundary condition is that the pull-up characteristic must be such that the setup time to rising APIC clock edge is met. For more information, consult the latest "Pentium® Processor Specification Update" on the APIC Bus.

Part II:

Specification Update for

82375 EB/SB PCEB

GENERAL INFORMATION

This section covers the 82375 EB/SB PCEB.

Component Markings

82375 EB/SB PCEB COMPONENT MARKING INFORMATION

Stepping	S-Spec	Top Marking	Freq.	Notes
A-2	SZ865	S 82375EB, SZ865	33	No longer available
B-0	SZ891	S 82375SB, SZ865	33	No longer available
B-1		S 82375SB	33	Production

Summary Table of Changes

The following table indicates the Specification Changes, Errata, Specification Clarifications, or Documentation Changes which apply to the listed 82375EB/SB PCEB. Intel intends to correct some of the errata in a future stepping of the component, and to account for the other outstanding issues through documentation or specification changes as noted. Any items that are shaded are new for this revision of the document. This table uses the following notations:

CODES USED IN SUMMARY TABLE

X:	Erratum, Specification Change or Clarification that applies to this stepping.
Doc:	Document change or update that will be implemented.
Fix:	This erratum is intended to be fixed in a future stepping of the component.
Fixed:	This erratum has been previously fixed.
NoFix	There are no plans to fix this erratum.
(No mark) or (Blank Box):	This erratum is fixed in listed stepping or specification change does not apply to listed stepping.
Shaded:	This item is either new or modified from the previous version of the document.

82375EB/SB PCEB

NO.	A2	B0	B1	Plans	SPECIFICATION CHANGES	Documentation Update Status
1	X			Doc-NoFix	Signals Requiring Pull-Ups.	
2	X	X	X	Doc-NoFix	External PCI Arbiter Not Supported.	
3	X	X	X	Doc	Icc Specification Defined.	
NO.	**A2**	**B0**	**B1**	**Plans**	**ERRATA**	**Documentation Update Status**
1	X	X	X	Doc-NoFix	LB Errors When MLT Expires and PCEB is Disconnected on PCI.	
2	X	X	X	Doc-NoFix	Incorrect Generation of PERR#.	
3	X			Doc	Deadlock in GAT Mode.	
4	X	X	X	Doc-NoFix	DP Mode : AFLUSH# Errata.	
5	X	X	X	Doc-NoFix	Semi-Subtractive Decoding of Interrupt Acknowledge.	
6	X	X	X	Doc-NoFix	EISA Data Setup to CMD# Asserted.	
7	X			Doc	PCEB Starvation Issue in PCI Priority Arbitration Mode F0h.	

NO.	A2	B0	B1	Plans	SPECIFICATION CLARIFICATIONS	Documentation Update Status
1	X	X	X	Doc	Reading & Writing Configuration Registers.	
2	X			Doc	Potential for Starvation Using Certain Arbitration Modes.	
3	X	X	X	Doc	Potential for Starvation When the MRT is Disabled.	
4	X	X	X	Doc	Ground Bounce on BE[3:0]	
5	X	X		Doc	FLUSHREQ# Protocol.	
6	X	X		Doc	EISA LB Under-run.	
NO.	**A2**	**B0**	**B1**	**Plans**	**DOCUMENTATION CHANGES**	**Documentation Update Status**
					There are currently no additional known documentation changes	
NO.	**A2**	**B0**	**B1**	**Plans**	**GENERAL CONSIDERATIONS**	**Documentation Update Status**
1					There are currently no known general considerations to include in this revision.	

82375EB/SB PCEB SPECIFICATION CHANGES

1. Signals Requiring External Pull-Ups

The following signals are tri-stated during PCI Reset:

GNT0#/PCEBREQ#, CPUGNT#, GNT1#/RESUME#, GNT[2:3]#, AFLUSH#.

Each of these signals requires an external pull-up resistor (8k-10kW).

2. External PCI Arbiter Not Supported

The 82375EB does not support an external PCI arbiter. System logic must ensure that CPUREQ# is sampled high on the rising edge of PCIRST#. The only exception to this specification change is when the EISA Bridge is implemented in conjunction with the Intel 440FX and 430HX North Bridge internal PCI arbiters.

3. ICC Specification Defined

4 EISA Slots : Icc = 170mA (max)

8 EISA Slots : Icc = 200mA (max)

82375EB/SB PCEB ERRATA

1. Line Buffer Errors When MLT Expires and PCEB is Disconnected on PCI

PROBLEM: When the PCEB is flushing its line buffers by performing a burst write transfer on PCI, a situation may occur where the PCEB is disconnected by the target device, and at the same time the Master Latency Timer (MLT) expires. The target device would have had to add wait states in order to allow the MLT to expire. When this situation occurs, the data belonging to the current data phase is also transferred in the next data phase. The data that should have been transferred is lost.

IMPLICATION: The system may malfunction.

WORKAROUND: Write F8h to the Master Latency Timer register (offset 0Dh). This guarantees that the MLT will not expire before the buffer flush has completed, even if the target device adds many wait states.

2. Incorrect Generation of PERR#

PROBLEM: An incorrect generation of PERR# by the PCEB may occur during a PCI to EISA I/O write cycle that gets retried. This may happen when the following conditions occur:

a. EISA to PCI line buffers are disabled or some EISA to PCI address regions are programmed with the line buffer attribute turned off for that region. This is non-standard usage of the PCEB, and may result in non-optimum performance.

b. Parity is enabled in the PCEB (register 04h, bit 6 =1).

WORKAROUND: Enable the EISA to PCI line buffers (default condition). If EISA to PCI Memory region address registers are used with parity enabled, then the memory region attributes should enable the line buffers for that region.

3. Deadlock in GAT Mode

PROBLEM: For EISA masters and DMA transfers, the ESC asserts NMFLUSH# to the PCEB and then tri-states its NMFLUSH# output driver on the following clock. The PCEB is supposed to hold NMFLUSH# asserted until all its buffers are flushed and MEMACK# has been returned by the host bridge. The PCEB then deasserts NMFLUSH#. After sampling NMFLUSH# negated, the ESC resumes driving NMFLUSH# and returns MACK# or DACK# to the requesting device. However, if the PCEBs line buffers are not empty and is programmed to GAT (Guaranteed Access Time) mode, the PCEB will fail to continue to assert NMFLUSH# thus driving NMFLUSH# de-asserted and EISAHLDA active without waiting for MEMACK# being driven active by the Host Bridge (PCMC). This problem has been fixed in the PCEB B-0 and B-1 steppings.

IMPLICATION: This condition causes the ESC to issue a DACK# or MACK# without the MEMACK# being asserted from the host bridge (the assertion of MEMACK# by the host bridge signifies that all data in the CPU-to-PCI write buffers has been flushed and further posting disabled). When the PCEB starts a transaction on the PCI Bus (acting on behalf of the EISA master) the PCMC will retry the PCEB if there is still some data in its CPU-to-PCI posted write buffer and would try to flush the data by acquiring the PCI bus. If the posted data in PCMCs CPU-to-PCI write buffer is targeted towards the EISA bus then the PCEB would retry the PCMC (since the EISA bus is already occupied) and a condition would occur in the system where both the PCMC and PCEB are retrying each other without either completing the cycle.

WORKAROUND I: Disable GAT mode by clearing bit 0 of the PCI Arbiter Control Register (offset 41h).

WORKAROUND II: This deadlock condition could be avoided by disabling the CPU-to-PCI write buffers.

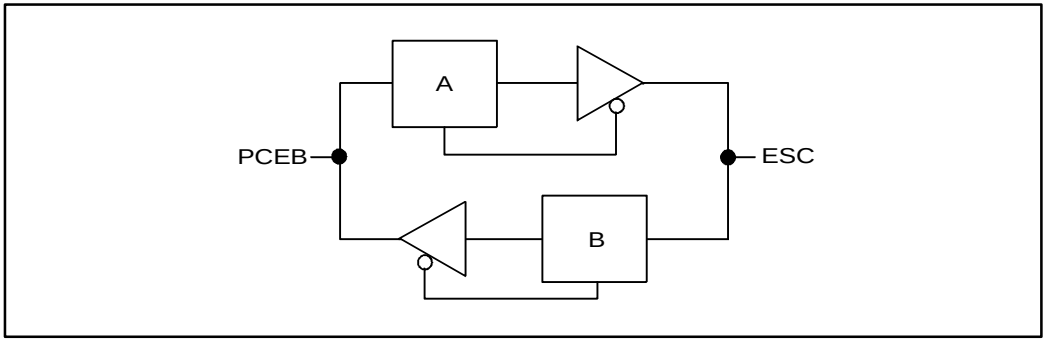
4. DP Mode: AFLUSH#

PROBLEM: In DP mode and during EISA/ISA/DMA transfers to PCI, an internal AC timing in the PCEB line buffers is violated, resulting in incorrect data driven onto the PCI bus. The failing scenario occurs during an APIC interrupt while the line buffers are flushing data and at the same time filling data. During that time, there is a window where the line buffer pointers do not change correctly.

IMPLICATION: In DP mode EISA/ISA/DMA transfers to PCI can incur data corruption. The problem can occur in both GAT and non-GAT mode under DP operation when the I/O APIC is enabled.

WORKAROUND: There are two possible workarounds to this erratum. The first workaround is to disable the EISA-to-PCI Line Buffers in the PCEB. However, disabling the PCEB line buffers may result in some add-in card incompatibilities.

The second workaround is designed to intercept the AFLUSH# signal between the PCEB and ESC (see figure below) in order to temporarily inhibit the assertion of the AFLUSH# signal to the PCEB. The AFLUSH# signal is held negated if there is an EISA-to-PCI transaction taking place. Once EISAHLDA is negated, the workaround will initiate the AFLUSH# protocol between the ESC and PCEB which will allow the APIC interrupt message to be sent to the processors. The workaround employs two state machines (using the PCI clock) that emulate the AFLUSH# logic from the both the PCEB and ESC perspectives.



AFLUSH# PLD Workaround: Inserted Between PCEB and ESC AFLUSH# Signal

The following state diagrams describe the PLD workaround. State machine A emulates the PCEB driving the ESC AFLUSH# pin. State Machine B emulates the ESC driving the PCEB AFLUSH# pin.

The sequence of events is as follows:

a) State Machine A (SMA) waits for the AFLUSH# signal from the ESC

b) Once the ESC drives AFLUSH#, SMA will assert and hold AFLSH_ESC#

c) State Machine B (SMB) waits for the bus via EISAHLDA

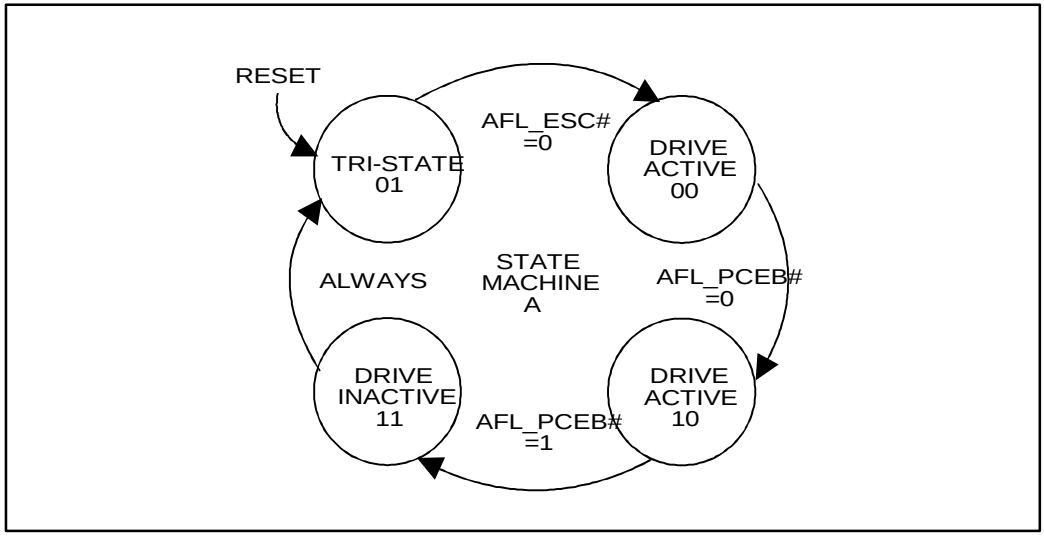
d) Once EISAHLDA is 0 (bus not busy) SMB asserts AFL_PCEB# for 1 clock

e) The PCEB assert AFLUSH# until buffers are flushed

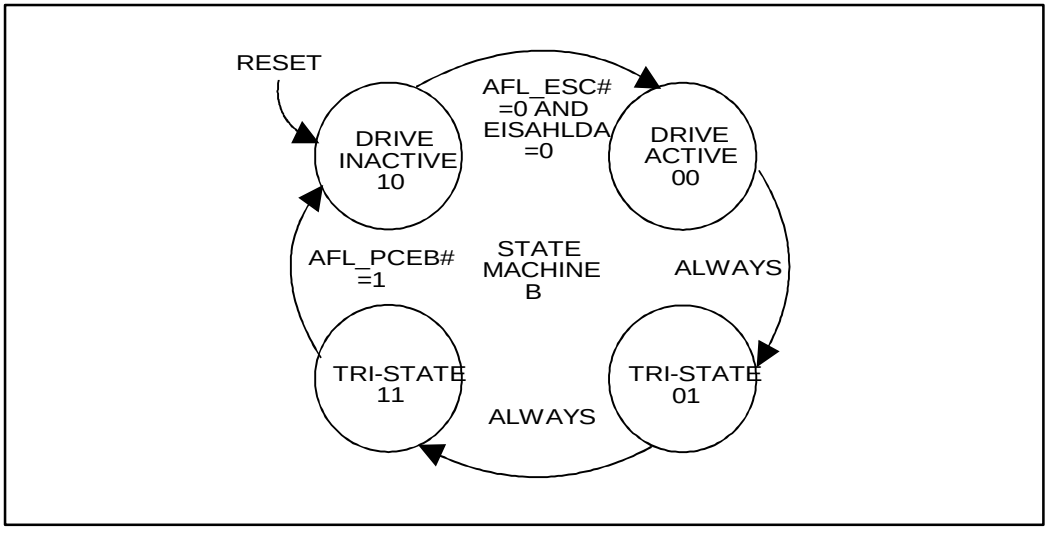
f) The PCEB signals the buffers are flushed by negating AFLUSH#

g) SMA drives AFLSH_ESC# inactive, then tri-states

h) SMB comes out of tri-state and negates AFLSH_PCEB# which is the default position for the ESC



PCEB Side of AFLUSH# which drives AFLSH_ESC#



State Machine B: ESC Side of AFLUSH# which drives AFLSH_PCEB#

The "E0320" device is a 20 pin 85C220-66 PLD. The PCI clock drives the PLD CLK input.

A soft copy of this .ABL file is available from Intel.

```
module  U1BXJR
title   'AFLUSH# fix for PCEB A-2 Errata #1 and #2
        and PCEB B-0 AFLUSH# Errata
        Rev. 0.0  Jan 31, 1994
        Author: Intel PCD'
U1BXJR    device  'E0320';

CLK                          pin 1;
EISAHLDA, !PCIRST_B          pin 3,4;
AFLSH_ESC_OE                 pin 19;

!AFLUSH_B_ESC, !AFLUSH_B_PCEB   pin 17,18;
QA1, QA0, QB1, QB0              pin 13,14,15,16;

"State machine A register assignment
srega      = [ QA1, QA0];
STAA       = [ 0,1 ];
STAB       = [ 0,0 ];
STAC       = [ 1,0 ];
STAD       = [ 1,1 ];

"State machine B register assignment
sregb      = [ QB1, QB0];
STBA       = [ 1,0 ];
STBB       = [ 0,0 ];
STBC       = [ 0,1 ];
STBD       = [ 1,1 ];

H,L,C,X,Z,P = 1,0,.C.,.X.,.Z.,.P.;

QA0,QA1,QB0,QB1  istype 'reg';

equations

[QA0,QA1].CLK = CLK;
[QB0,QB1].CLK = CLK;
```

```
AFLUSH_B_ESC = !QA0.FB & !PCIRST_B;
AFLSH_ESC_OE = !(!QA1.FB & QA0.FB);
AFLUSH_B_ESC.OE = AFLSH_ESC_OE;

AFLUSH_B_PCEB = !QB1.FB & !PCIRST_B;
AFLUSH_B_PCEB.OE = !QB0.FB;

State_diagram srega
State STAA:
        IF (PCIRST_B) THEN STAA
        ELSE
          IF (AFLUSH_B_ESC) THEN STAB
          ELSE STAA;

State STAB:
        IF (PCIRST_B) THEN STAA
        ELSE
          IF (AFLUSH_B_PCEB) THEN STAC
          ELSE STAB;

State STAC:
        IF (PCIRST_B) THEN  STAA
        ELSE
          IF (!AFLUSH_B_PCEB) THEN STAD
          ELSE STAC;

State STAD:
        GOTO STAA;
```

5. *Semi-Subtractive Decoding of Interrupt Acknowledge Cycles*

PROBLEM: If the PCEB is programmed to the optional mode of decoding PCI interrupt acknowledge cycles in a semi-subtractive manner (PCICON register, address offset 40h, bit 5=0), the PCEB does not properly allow an external PCI-based interrupt controller to respond to an interrupt acknowledge cycle. The PCEB may also drive DEVSEL# and STOP# signals after the external PCI-based interrupt controller has claimed the interrupt acknowledge cycle.

IMPLICATION: Contention can result due to two devices driving the PCI bus.

WORKAROUND: Program the PCICON register, address offset 40h to set bit 5=1 (this is the default setting). This setting will enable the PCEB for positive decode of the interrupt acknowledge cycle (standard/default usage).

6. EISA Data Setup to CMD# Asserted

PROBLEM: Because of the partitioning between the PCEB and ESC, the PCEB drives data on to the EISA bus when it sees CMD# asserted from the ESC. The data valid delay period from CMD#, driven by the PCEB, causes two known EISA add-in cards to fail and does not meet the EISA Data setup to CMD* asserted timing for 32 bit EISA write cycles.

(The following is from the BCPR Specification Rev. 3.12)

(for Figure 81) 8,16 or 32-bit EISA Slave Timing (pgs. 199-200):

Parameter	Description	Min. (ns)
25	Data setup to CMD* asserted (write 32 bit slave)	-10

This describes the EARLIEST time when data is valid for a slave to latch it.

IMPLICATIONS: Most EISA cards latch data on the rising edge of CMD# toward the end of the EISA cycle. This happens long after (>100nS) the data driven by the PCEB becomes valid allowing most cards to function properly. The first known card affected by this issue is a very high-end EISA graphics card during configuration cycles. This card is no longer in production and has been discontinued by the manufacturer.

This issue also impacts an EISA Novell Mirrored Server Link (NMSL) card during IO configuration cycles. This card does not behave as most typical EISA cards in that it does expect data to be provided well before the rising edge of CMD#. The vendor for this particular card is working on modifying the board to latch the data on the rising edge of CMD# during I/O configuration cycles. The vendor is expected to have a solution soon.

WORKAROUND: Even though corrections are being made to the EISA card mentioned above, for those customers who require operation with the existing revision of the card, there is a PLD workaround. This PLD generates CMD# early to the PCEB so that the PCEB provides data within the required spec.

Here are the PLD equations (written in PLDshell Plus) for the data setup to CMD# workaround:

```
PIN 1  PCICLK        ; clk input
PIN 2  START         ; input
PIN 3  WRITE         ; input
PIN 4  EHLDA         ; input
PIN 5  NMFLUSH       ; input
PIN 6  CMD_ESC       ; input
PIN 7  BCLK          ; input
PIN 18 CMD_EN        ; input/output
PIN 19 CMD_PCEB      ; output to PCEB

EQUATIONS
        CMD_EN.CLKF = PCICLK
1)      CMD_EN.D := ( /EHLDA + /NMFLUSH ) * /BCLK * /START * WRITE
2)      CMD_PCEB = /CMD_EN * CMD_ESC
```

Notes:

1) This registered output generates a "high" signal during the first part of the I/O write cycle when the PCEB is in the process of flushing its buffers in preparation for ownership of the bus. It ensures that the bus is owned by the PCEB.

2) CMD_EN is inverted since CMD# is active-low. A logical AND operation with the CMD# driven by the ESC generates the early CMD_PCEB signal to the PCEB. Since this allows the PCEB to sense CMD# "low" BEFORE the ESC drives CMD#, it will drive out data earlier.

7. PCEB Issue In PCI Priority Arbitration Mode F0h

PROBLEM: When the PCI Arbiter Priority Control Register (offset 42h) is programmed for purely rotational priority mode (F0h), the PCEB can starve itself if Bank3 (REQ1# and REQ2#) is inactive. When REQ1# and REQ2# are unused or not requesting the bus, the arbitration rotation halts on Bank3. This essentially forces the arbiter to a fixed arbitration mode having CPUREQ# and REQ3# with the higher priority.

IMPLICATION: If REQ1# and REQ2# are inactive, the rotational priority scheme will be forced to a fixed mode introducing the potential for the PCEB to never grant itself the bus because of the two PCI masters (CPUREQ# and REQ3#) having higher priority (see Specification Clarification #2, Potential for Starvation Using Certain Arbitration Modes).

WORKAROUND: Do not use REQ3#, or program the PCI Arbiter Priority Control Register (offset 42h) to a fixed priority mode which does not conflict with Specification Clarification #2.

82375EB/SB PCEB SPECIFICATION CLARIFICATIONS

1. Reading and Writing Configuration Registers

When changing values in configuration registers which contain one or more reserved bits, care must be taken to avoid changing the value of the reserved bits. It is not safe to assume that because a bit is reserved that it has no function or meaning. The proper procedure for changing a register value is to first read the register, change only the required bits (without changing the value of any reserved bits), then write the new value back out to the register.

2. Potential Issue Using Certain Arbitration Modes

If two or more PCI masters have a higher priority than the PCEB in a fixed arbitration mode (modes 04h, 05h, 06h, 07h, 08h, 09h, 0Ah, 0Bh), then the possibility exists that the PCEB will never grant itself the bus due to heavy PCI traffic by the two or more PCI masters, and the PCEB may starve itself. When using one of the fixed arbitration modes, do not have more than one PCI master with a priority above that of the PCEB. For example, if arbitration mode 4 is selected, do not use REQ3#.

3. Potential Issue When the Master Retry Timer is Disabled

If the Master Retry Timer is disabled, the PCEB may never grant itself the PCI bus, and the PCEB may starve itself. The solution is to always program the Master Retry Timer (register 41 bits [4:3]) for 16, 32, or 64 PCICLKs.

4. Ground Bounce On BE[3:0]

Ground bounce may occur on the PCEB's Byte Enable (BE1#, BE3#) signals. This only occurs when any of the PCEB's SD[7:0] data lines transition from V_{OH} to V_{OL} and when the EISA Bridge is Reading/Writing out to the EISA Bus. EISA Master Reads/Writes are not affected by this issue. The Byte Enable Bus (BE[3:0]#) and the lower byte of the System Data Bus (SD[7:0]) are related because they share an internal ground. Whenever the SD Bus dumps large amounts of current to ground, the SD Bus tends to exhibit predatory behavior upon the much smaller BE[3:0]# Bus and may cause ground bounce in the BEX# signals.

The Byte Enable Signals (BE[3:0]#) are responsible for determining data steering and determines which of the four data byte lane(s) are active. The ground bounce on the BEX# Bus has the potential to cause data corruption. However, the SD Bus and the BE[3:0]# Bus only host synchronous signals. This means that in almost all current systems, the ground bounce problem is occurring between clocks and therefore is a "don't care" situation. The BE[3:0]# "Hold time from BCLK rising" specification is a minimum of 20nS and the BEX# ground bounce occurs at approximately the 18nS mark. This ground bounce issue is only known to affect one EISA card manufacturer and only then because this particular manufacturer's EISA card samples the Byte Enable lines towards the latter end of the BE[3:0]# Bus "Hold from BCLK" timing specification.

A single "capacitor to ground" located as closely as possible to BE1# and BE3# is recommended. Suggested value range of the capacitor is approximately 50pF - 68pF. Many I/O Card manufacturers need not specify any design changes due to the position of the ground bounce in an EISA slave cycle.

5. FLUSHREQ# Protocol

A livelock scenario can exist if the PCEB contained data within its line buffers pointed towards the host bridge, and at the same time the host bridge contained data pointed towards the PCEB. The PCEB B-1 has changed the FLUSHREQ# protocol to accommodate these host bridges.

6. EISA Line Buffer Under-run

If an EISA master is reading from memory, and the PCI slave (host bridge) inserts more wait states such that each PCI data transfer is slower than EISA, incorrect data can be forwarded on the bus. The PCEB B-1 line buffer logic has been corrected to avoid this

82375EB/SB PCEB DOCUMENTATION CHANGES

There are currently no additional known documentation changes to include in this revision.

82375 EB/SB PCEB GENERAL CONSIDERATIONS

There are currently no known general considerations to include in this revision.

82375SB (PCEB) B-1 STEPPING CHANGES FOR INTEL HOST BRIDGES OTHER THAN THE INTEL PCMC.

The PCEB is designed and validated for use with Intel host bridges only. Intel does not guarantee the operation or functionality of the PCEB with non-Intel host bridges.